

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<small>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</small> PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 20-08-2004		2. REPORT TYPE Technical Report		3. DATES COVERED (From - To) Jun 2003-Aug 2004	
4. TITLE AND SUBTITLE Interior-point algorithms, penalty methods and equilibrium problems			5a. CONTRACT NUMBER		
			5b. GRANT NUMBER N00014-04-1-0145		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Benson, Hande, Y Sen, Arun Shanno, David, F Vanderbei, Robert, J			5d. PROJECT NUMBER		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104-2875 Princeton University, Princeton, NJ 08544 RUTCOR - Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854-8003			8. PERFORMING ORGANIZATION REPORT NUMBER ORFE-03-02		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research, Ballston Centre Tower One, 800 North Quincy Street, Arlington, VA 22217-5660 National Science Foundation, 4201 Wilson Boulevard, Arlington, VA 22230			10. SPONSOR/MONITOR'S ACRONYM(S) ONR, NSF		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited					
13. SUPPLEMENTARY NOTES Submitted to Computational Optimization and Applications			20050111 092		
14. ABSTRACT In this paper we consider the question of solving equilibrium problems—formulated as complementarity problems and, more generally, mathematical programs with equilibrium constraints (MPECs)—as nonlinear programs, using an interior-point approach. These problems pose theoretical difficulties for nonlinear solvers, including interior-point methods. We examine the use of penalty methods to get around these difficulties, present an example from game theory where this makes a difference in practice, and provide substantial numerical results. We go on to show that penalty methods can resolve some problems that interior-point algorithms encounter in general.					
15. SUBJECT TERMS interior-point methods, nonlinear programming, penalty methods, equilibrium problems, complementarity					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Unclassified Unlimited	18. NUMBER OF PAGES 38	19a. NAME OF RESPONSIBLE PERSON Hande Y. Benson
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) 215-895-6999

INTERIOR-POINT ALGORITHMS, PENALTY METHODS AND EQUILIBRIUM PROBLEMS

HANDE Y. BENSON, ARUN SEN, DAVID F. SHANNO, AND ROBERT J. VANDERBEI

ABSTRACT. In this paper we consider the question of solving equilibrium problems—formulated as complementarity problems and, more generally, mathematical programs with equilibrium constraints (MPEC's)—as nonlinear programs, using an interior-point approach. These problems pose theoretical difficulties for nonlinear solvers, including interior-point methods. We examine the use of penalty methods to get around these difficulties, present an example from game theory where this makes a difference in practice, and provide substantial numerical results. We go on to show that penalty methods can resolve some problems that interior-point algorithms encounter in general.

1. INTRODUCTION

Devising reliable ways to find equilibria of dynamical systems has attracted much interest in recent years. Many problems in mechanics involve computing equilibria, as do problems in economics dealing with general equilibrium or game theory. In finance certain options pricing problems may also be formulated as equilibrium models. What is of interest here is that equilibrium problems can be viewed in a natural way as mathematical programming problems. In this context, they are usually referred to as complementarity problems. Very briefly, a complementarity problem is to find a vector

$$x \in \mathbb{R}^n \quad \text{s.t.} \quad 0 \leq x \perp g(x) \geq 0,$$

for a function $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$, and $x \perp g(x)$ means $x^T g(x) = 0$.

Date: August 20, 2004.

1991 Mathematics Subject Classification. 90C51, 90C31, 90C33, 90C90.

Key words and phrases. interior-point methods, nonlinear programming, penalty methods, equilibrium problems, complementarity.

Research of the first author is sponsored by ONR grant N00014-04-1-0145. Research of the second and fourth authors is sponsored by NSF grant DMS-9870317 and ONR grant N00014-98-1-0036. Research of the third author is supported by NSF grant DMS-0107450.

For a survey of common applications of complementarity problems, see [7]. The options pricing example is described in [12]. For a survey of common algorithmic approaches to solving these problems, see [8]. Most of the methods in [8] are designed specially for solving complementarity problems. Many of them involve solving a system of non-smooth equations, while some of them use a simplex-type pivoting tool. By and large, much of the work involving these methods has been theoretical; reliable practical implementations have been rare. Thus many real-world equilibrium problems have remained intractable, leaving room for new approaches.

Recently one popular approach has been to view complementarity problems as a type of nonlinear program. This idea had not been regarded very favorably in the past, since nonlinear codes were not very robust, and equilibrium problems pose theoretical difficulties for nonlinear programming. However, the advent of reliable nonlinear codes in recent years has made it possible to reconsider this approach. Moreover, the use of general nonlinear solvers makes it possible to add an objective function to the problem, thus bringing under one framework, complementarity problems and the more general mathematical problems with equilibrium constraints (MPEC's).

The rest of this paper is organized as follows: In Section 2, a general interior-point code for nonlinear programming is described. In Section 3, we get into the details of complementarity problems and MPEC's, and the theoretical difficulties they might pose for nonlinear solvers. Section 4 describes a way to resolve these difficulties, namely penalty methods (also known as constraint relaxation). Section 5 gives an example from game theory which highlights why these theoretical considerations can actually matter. Section 6 describes an implementation of a penalty method using LOQO, and describes numerical results for two standard testsets using this implementation. In Section 7 we show that penalty methods can resolve two problems that interior-point methods such as LOQO can encounter in general (not just in the context of MPEC's): jamming and infeasibility detection.

2. AN INTERIOR-POINT METHOD: A BRIEF DESCRIPTION

The following is only a brief description of an interior-point method for nonlinear programming, for details, see LOQO [20] or IPOPT [21]. Another interior-point method, KNITRO [4] uses the same paradigm, but its implementation is very different. For ease of notation we describe the algorithm for problems with inequality constraints only (the situation is much the same for equality constraints—again, see [20]).

Thus the problem is

$$(1) \quad \begin{array}{ll} \min & f(x) \\ \text{s.t.} & h_i(x) \geq 0, \quad i = 1, \dots, m, \end{array}$$

where we assume that $f(x)$ and each of the $h_i(x)$ are twice continuously differentiable and $x \in \mathbb{R}^n$. Adding nonnegative slacks, w , to the inequality constraints in (1) and putting these slacks in a logarithmic barrier term in the objective function, the problem becomes:

$$(2) \quad \begin{array}{ll} \min & f(x) - \theta \sum_{i=1}^m \log(w_i) \\ \text{s.t.} & h(x) - w = 0. \end{array}$$

Letting the vector λ represent the Lagrange multipliers for (1) the first-order optimality conditions for the problem are

$$(3) \quad \begin{array}{rcl} \nabla f(x) - A(x)^T \lambda & = & 0, \\ -\theta e + W \Lambda e & = & 0, \\ h(x) - w & = & 0. \end{array}$$

Here e is the vector of all ones, $A(x)$ is the Jacobian of the vector $h(x)$, and W and Λ are the diagonal matrices with the coordinates of the vectors w and λ on the diagonals, respectively.

We use one step of Newton's method to generate a triple (x, w, λ) which solve the linearization of (3) for each value of θ , where the sequence of θ 's monotonically decreases to 0 [20] as we approach the optimal solution. We set

$$\begin{aligned} H(x, \lambda) &= \nabla^2 f(x) - \sum_{i=1}^m \lambda_i \nabla^2 h_i(x), \\ \sigma &= \nabla f(x) - A(x)^T \lambda, \\ \gamma &= \theta W^{-1} e - \Lambda, \\ \rho &= w - h(x). \end{aligned}$$

We then find the directions given by Newton's method by solving the *reduced* KKT system:

$$\begin{bmatrix} -H(x, \lambda) & A(x)^T \\ A(x) & W \Lambda^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \sigma \\ \rho + W \Lambda^{-1} \gamma \end{bmatrix}.$$

Δw is computed according to:

$$\Delta w = W \Lambda^{-1} (\gamma - \Delta \lambda).$$

If the algorithm is currently at the point (x^k, w^k, λ^k) , the new point $(x^{k+1}, w^{k+1}, \lambda^{k+1})$ is obtained as follows: Let

$$T(x, w, \lambda) = \begin{bmatrix} x \\ w \\ \lambda \end{bmatrix} + \alpha(x, w, \lambda) \begin{bmatrix} \Delta x(x, w, \lambda) \\ \Delta w(x, w, \lambda) \\ \Delta \lambda(x, w, \lambda) \end{bmatrix}$$

Then

$$\begin{bmatrix} x^{k+1} \\ w^{k+1} \\ \lambda^{k+1} \end{bmatrix} = T(x^k, w^k, \lambda^k)$$

where α is chosen to ensure that $w^{k+1} > 0$, $\lambda^{k+1} > 0$, and that either the barrier function or the infeasibility is sufficiently reduced without making the other much worse (see [3] for details). Note that this ensures that multipliers corresponding to inequality constraints will always remain positive, a point that will be relevant later.

LOQO decides that a solution is optimal when the primal infeasibility, ρ , the dual infeasibility, σ , and the duality gap, $\frac{w^T \lambda}{m}$, are sufficiently small.

3. COMPLEMENTARITY PROBLEMS AND MPEC'S

As mentioned in the introduction, the complementarity problem is to find a vector

$$x \in \mathbb{R}^n \quad \text{s.t.} \quad 0 \leq x \perp g(x) \geq 0,$$

for a function $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$, and $x \perp g(x)$ means $x^T g(x) = 0$. Note that the nonnegativity also implies $x_i g_i(x) = 0, i = 1, \dots, n$. We will refer to this way of writing it as the *pairwise* formulation (and the first way as the *summation* formulation). For now we will use the pairwise formulation, but the two formulations can have differing effects on algorithmic performance (more on this later).

The more general *mathematical program with equilibrium constraints* (MPEC) may be written as the NLP

$$(4) \quad \begin{array}{ll} \min & f(x) \\ \text{s.t.} & h_i(x) = 0, i = 1, \dots, m, y_i = g_i(x), i = 1 \dots n, \\ & x, y \geq 0, x_i y_i \leq 0, i = 1, \dots, n, \\ & x = (x_1, x_2) \in \mathbb{R}^{n+l}, g : \mathbb{R}^{n+l} \rightarrow \mathbb{R}^n. \end{array}$$

The first question which arises is, why does this problem deserve special attention? The reason is that (4) has a special structure which is guaranteed to cause theoretical difficulties. Specifically, it is a well-known fact that the set of optimal Lagrange multiplier solutions is always unbounded (see, for instance, [15]), when a standard constraint

qualification, known as MPEC-LICQ, holds (MPEC-LICQ simply says that standard NLP constraint qualifications are violated only because of the complementarity constraints- see [15] for a formal definition). It is easy to see that this remains true for the multipliers that are generated when applying an interior-point method. The next thing to realize is that an unbounded set of Lagrange multipliers at the optimum can cause problems for many traditional nonlinear solvers, and especially for an interior-point code. This is due to the fact that interior-point methods try to find the center of the face of optimal solutions for any set of variables (primal or dual). Thus an unbounded set of optimal multipliers can mean that the multipliers get larger and larger as a solution is approached. This in turn can cause serious problems for the algorithm, as seen by simply noting that one of the equations in the KKT system (3) is

$$W\Lambda = \theta e,$$

which means that as multipliers get large, slacks must get small. If the multipliers become large while θ is still significantly greater than 0, then slacks might have to become small prematurely. Since the slacks are always positive, very small slacks can result in excessively shortened steplengths, slowing progress of the algorithm, or making it stall completely.

The problem with unbounded multipliers just described exists even if every pair in a solution is *strictly* complementary (meaning that exactly one member of each pair is 0). For interior-point methods the problem is compounded when the solution is not strict. We can see this as follows:

We set up the interior-point paradigm for (4), by adding nonnegative slacks to the inequality constraints, and putting them in a barrier term in the objective function:

$$(5) \quad \begin{aligned} \min \quad & f(x) - \theta \sum_{i=1}^n \log w_i \\ \text{s.t.} \quad & h_i(x) = 0, i = 1, \dots, m, y_i = g_i(x), i = 1 \dots n, \\ & x, y \geq 0, x_{1i}y_i(x) + w_i = 0, i = 1, \dots, n. \end{aligned}$$

We then form the Lagrangian for this problem:

$$\begin{aligned} \mathcal{L}(x, y, w, \lambda, \mu) = & f(x) + \sum_{i=1}^n \lambda_{1i}(x_{1i}y_i(x) + w_i) - \sum_{i=1}^n \mu_{1i}x_{1i} - \\ & \sum_{i=1}^n \mu_{yi}y_i - \sum_{i=1}^l \mu_{2i}x_{2i} + \sum_{i=1}^n \lambda_{2i}h_i(x) + \sum_{i=1}^n \lambda_{3i}(y_i - g_i(x)) - \theta \sum_{i=1}^n \log w_i, \end{aligned}$$

where (λ, μ) are of course the vectors of Lagrange multipliers. Now if a given solution x^* to (5) is strictly complementary, meaning that in each complementary pair exactly one of x_{1i}^* and y_i^* is 0 (and the other strictly positive), we may do the following: Partition the set $\{1, \dots, n\}$ into two sets I and J , with J representing those indices j for which x_{1j}^* is 0, and I those i for which y_i^* is 0. We use the notation (standard for interior-point methods) that a capitalized variable name represents a diagonal matrix whose elements correspond to the vector with the same lower-case name. The KKT conditions are as follows:

$$\begin{aligned}
 (6) \quad \nabla_{x_1} \mathcal{L} &= \begin{bmatrix} \nabla_{x_{1I}} f(x^*) \\ \nabla_{x_{1J}} f(x^*) \end{bmatrix} + \begin{bmatrix} 0 \\ y_J^{*T} \end{bmatrix} \begin{bmatrix} \lambda_{1I}^* \\ \lambda_{1J}^* \end{bmatrix} - \\
 &\quad \begin{bmatrix} \mu_{1I}^* \\ \mu_{1J}^* \end{bmatrix} + \begin{bmatrix} \nabla_{x_{1I}} h(x^*) \\ \nabla_{x_{1J}} h(x^*) \end{bmatrix} \begin{bmatrix} \lambda_{2I}^* \\ \lambda_{2J}^* \end{bmatrix} - \begin{bmatrix} \nabla_{x_{1I}} g(x^*) \\ \nabla_{x_{1J}} g(x^*) \end{bmatrix} \begin{bmatrix} \lambda_{3I}^* \\ \lambda_{3J}^* \end{bmatrix} = 0, \\
 \nabla_y \mathcal{L} &= \begin{bmatrix} x_{1I}^{*T} \\ 0 \end{bmatrix} \begin{bmatrix} \lambda_{1I}^* \\ \lambda_{1J}^* \end{bmatrix} + \begin{bmatrix} \lambda_{3I}^* \\ \lambda_{3J}^* \end{bmatrix} - \begin{bmatrix} \mu_{yI}^* \\ \mu_{yJ}^* \end{bmatrix} = 0, \\
 \nabla_{x_2} \mathcal{L} &= \nabla_{x_2} f(x^*) - \mu_2^* + \nabla_{x_2} h(x^*) \lambda_2^* - \nabla_{x_2} g(x^*) \lambda_3^* = 0. \\
 \frac{\partial \mathcal{L}}{\partial w_i} &= 0 \text{ implies } \lambda_{1i}^* w_i = \theta, \quad i = 1, \dots, n.
 \end{aligned}$$

In addition, there are of course primal feasibility and the usual complementarity conditions for the multipliers μ . The algorithm attempts to get to a solution to this system with $\theta = 0$.

If x^* is a non-strict complementary solution, i.e. for some i , $x_{1i}^* = y_i(x^*) = 0$, then by inspection of the KKT system we see that the corresponding element of λ_1^* drops out and has no effect on the system. This results in a rank deficient system, which can cause serious problems for nonlinear solvers, in particular for interior-point methods (see the discussion of numerical results in Section 6).

Despite the theoretical difficulties, unboundedness of the optimal multiplier set has thus far not usually been a serious problem in practice. Indeed, a fair amount of success has recently been achieved in solving MPEC's as nonlinear programs. LOQO has proved quite successful in solving a large number of MPEC's directly as nonlinear programs, as have SQP methods (see [8]). This could lead to the conclusion that MPEC's need not be regarded as a special problem class, and particular methods for dealing with them are largely unnecessary. In Section 5, however, we present a potentially important example which is not solved robustly if handled as an ordinary NLP. Methods which bound the optimal multipliers, on the other hand, do solve it successfully. These methods we describe in the next Section.

We conclude this Section with one final observation which will be important later: assuming MPEC-LICQ, it is not hard to show that the KKT system (6) has Lagrange multiplier solutions for any x^* which satisfies the complementarity constraints strictly, in other words the objective function f has no role in determining whether x^* is a solution. That is, every feasible point is a KKT point, for any f , if one assumes strict complementarity. This implies that the feasible set is discrete. Note that this observation only applies to problems in which all variables are involved in the complementarity constraints.

4. PENALTY METHODS

Consider again our NLP in the form

$$(7) \quad \begin{array}{ll} \min & f(x) \\ \text{s.t.} & h_i(x) = 0, i = 1 \dots m, y_i = g_i(x), i = 1 \dots n, \\ & x, y \geq 0, x_i y_i(x) \leq 0, i = 1 \dots n. \end{array}$$

A common approach in the past to deal with difficult constraints in an NLP has been to replace them with a penalty term in the objective function, which one tries to drive to zero. That is, suppose one has the problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & h(x) = 0. \end{array}$$

This is replaced with the penalized problem

$$\min f(x) + \rho P(x).$$

$P(x)$ is usually a non-negative function which equals 0 precisely when $h(x) = 0$. The penalty parameter ρ is a positive constant. The hope is that making ρ large enough will produce a solution to the penalized problem with $P(x) = 0$ (and hence a solution to the original problem). This is only possible if a solution to the original problem is a solution to the penalized problem. In fact, ideally every solution to the true problem would be a solution for the penalized problem. This does not have to hold in general. The most standard penalty function, the quadratic (which sets $P(x) = \sum_i (h_i(x))^2$) does not have this property. Penalty functions which do have this property are called exact.

4.1. Exact Penalty Functions. The two most common exact penalty functions are the ℓ_1 penalty function, defined by $P(x) = \sum_i |h_i(x)|$, and the ℓ_∞ , defined by $P(x) = \max_i \{|h_i(x)|\}$.

The exactness property guarantees that if one of these penalty functions is used, then every solution of the true problem is a solution to the penalized problem for sufficiently large (but finite) penalty ρ . Thus,

if x^* is a solution to the true problem, then picking a starting point sufficiently close to x^* when solving the penalized problem guarantees convergence to x^* (i.e. local convergence). This desirable property of the ℓ_1 and the ℓ_∞ penalty functions is generally proved assuming constraint qualification conditions hold at a solution to the true problem. As such, penalty methods (i.e. constraint relaxation) have been applied until now to problems that are assumed to be well-behaved (see e.g. [10]). MPEC's of course are not well-behaved, but we shall see that this is not a problem.

First let us see what the ℓ_1 and ℓ_∞ penalty functions look like when applied to the complementarity constraints in (4):

For the ℓ_1 norm we have the penalized problem

$$\begin{aligned} \min \quad & f(x) + \rho \sum |x_{1i}y_i(x)| \\ \text{s.t.} \quad & x, y \geq 0, y = g(x), h(x) = 0, \end{aligned}$$

which, because x and $g(x)$ are constrained to be non-negative, is equivalent to

$$(8) \quad \begin{aligned} \min \quad & f(x) + \rho \sum x_{1i}y_i(x) \\ \text{s.t.} \quad & x, y \geq 0, y = g(x), h(x) = 0, \end{aligned}$$

which may also be written as

$$(9) \quad \begin{aligned} \min \quad & f(x) + \rho \sum \zeta_i \\ \text{s.t.} \quad & x_{1i}y_i(x) \leq \zeta_i, \quad i = 1, \dots, n, \\ & x, y \geq 0, y = g(x), h(x) = 0. \end{aligned}$$

For the ℓ_∞ norm we have

$$\begin{aligned} \min \quad & f(x) + \rho \max_i \{|x_{1i}y_i(x)|\} \\ \text{s.t.} \quad & x, y \geq 0, y = g(x), h(x) = 0, \end{aligned}$$

which can be written as

$$(10) \quad \begin{aligned} \min \quad & f(x) + \rho \zeta \\ \text{s.t.} \quad & x_{1i}y_i(x) \leq \zeta, \quad i = 1, \dots, n, \\ & x, y \geq 0, y = g(x), h(x) = 0. \end{aligned}$$

Note that (9) and (10) only involve differentiable functions, so normal optimization methods can be used.

It is shown in, among other places, [1] that any solution to (4) is a solution to (10) for large enough (finite) ρ , in other words the ℓ_∞ penalty function is still exact, if (4) satisfies MPEC-LICQ. What is more, the result in [1] implies that any solution to (4) is a solution, for

a large enough but finite penalty, to the problem

$$(11) \quad \begin{aligned} \min \quad & f(x) + \rho\zeta \\ \text{s.t.} \quad & \sum_{i=1}^n x_i y_i(x) \leq \zeta, i = 1 \dots n, \\ & x, y \geq 0, y = g(x), h(x) = 0, \end{aligned}$$

i.e. what one gets if the original problem had been written with the *summation* formulation (see Section 3). And plainly (11) is equivalent to (8). Hence the ℓ_1 penalty function is also exact when applied to MPEC's (with MPEC-LICQ).

Thus these penalty methods could be an attractive way to solve MPEC's, if they took care of the original problem posed by MPEC's, namely that the optimal set of Lagrange multipliers is unbounded. And indeed, this is precisely what occurs. This is a well-known fact, and is shown in, for instance, [1]. It is trivial to show that the multipliers remain bounded when a LOQO-type interior-point method is applied to (10) (or (9)). For completeness we now state this as a theorem, since it will be crucial to all the work that follows.

Theorem 1. *If an interior-point method (as described in section 2) is applied to (10), the set of optimal Lagrange multipliers for a strictly complementary solution is bounded, provided that (4) satisfies MPEC-LICQ.*

In the case of a solution to (10) being a non-strict complementary solution to (4), this time there is no difficulty, the reason being that part of the KKT conditions to (10) is the equation:

$$(12) \quad \rho = \sum_{i=1}^n \lambda_i,$$

where the λ 's are the multipliers corresponding to the complementarity constraints. (12) ensures that none of the multipliers will drop out of the system. One further point to note is that if the original problem does not satisfy MPEC-LICQ, there might not be a KKT point (i.e. Lagrange multipliers might not exist). See Section 6 for an example of what happens in this case.

We now have two penalty methods, using the ℓ_1 and ℓ_∞ penalty functions, which are exact and keep the optimal multiplier set bounded, when applied to MPEC's satisfying standard constraint qualifications. They would therefore appear to be promising alternatives to solving an MPEC directly as an NLP. Some evidence for this already exists. In

[17], the ℓ_1 penalty method is applied to pure complementarity problems (actually mixed complementarity problems), and promising numerical results are obtained. LOQO is a more robust code, as it can deal with an indefinite Hessian matrix, as can happen in non-convex problems. In such cases LOQO adds a diagonal perturbation to the Hessian matrix H , making it $H + \lambda I$, where I is the identity matrix, and λ is a positive scalar. A large enough λ makes $H + \lambda I$ positive definite, and the algorithm may proceed successfully with this modified KKT matrix (see [20]). One thus expects that LOQO might be quite successful at using penalty methods to solve MPEC's. To this end, we modified the interior-point code of LOQO to incorporate complementarity constraints and use an ℓ_∞ penalty function to maintain boundedness of the dual variables. The main thrust of this paper is to test the effectiveness of this implementation. While this paper does not deal with an ℓ_1 penalty method for MPECs, Section 7 discusses a theoretical result that makes such an implementation of interest for general NLPs. Future work will report on this implementation.

5. AN EXAMPLE FROM GAME THEORY

We consider a model that is proposed in [18] and [19] as an extension of the model discussed extensively in those papers (and analyzed in [17]). That version of the model is mentioned in [5] as being traditionally regarded as a very hard equilibrium problem, so we regard the extended version of the model as a good challenge for our efforts. The AMPL model can be found at [16].

Very briefly, the model is of a game involving two (identical) firms which compete on the basis of price to sell an identical product. The game takes place over an infinite time-horizon at discrete intervals. In each round, a firm may or may not be free to move. If it is, it has the option to choose a price from a discrete set of prices or sit out of the market for a round. If it does choose a price, it is committed to that price for the following round as well. The payoff matrix for an individual round is determined by a standard duopoly structure (see [18] and [19]). Each firm tries to maximise its future (discounted) payoff. The derivation of the necessary and sufficient conditions for a (mixed-strategy) Nash equilibrium is similar to that in [18] and [19] for the simpler version of the model. These can then be formulated as an MPEC with a constant objective function along the lines of [17].

The action space of this problem is $\{-1, 0, 1, \dots, n-1\}$, where $0, \dots, n-1$ are the n prices in the model, and -1 corresponds to not choosing a price. The payoff matrix is Π , whose (i, j) entry gives

the payoff to a firm if it chooses action i , and its rival chooses action j . The decision variables of this problem consist of:

i) A reaction matrix R . The (i, k) , $i \geq 0$, entry of R gives the probability that the firm will take action k when its rival is committed to price i from the previous round.

ii) A simultaneous-strategy vector s . The entry k of s gives the probability that the firm takes action k when it does not know its rival's price for the current period.

iii) Payoff functions. These are:

v_{Ri} : the expected discounted payoff for a firm which is free to react to its rival's commitment to price i from the previous period

v_S : the expected discounted payoff for a firm when both it and its rival are free to choose new prices simultaneously for the current period

w_{Ri} : the expected discounted payoff for a firm when it is committed to price i from the previous period, but its rival is free to react

w_{Sij} : the expected discounted payoff for a firm when it is committed to price k , and its rival to price j from the previous period

The resulting MPEC is then

$$\begin{aligned}
 r_{ik}[v_{Ri} - (\Pi_{ki} + \delta \left\{ \begin{array}{l} w_{Rk} \text{ if } k \geq 0 \\ v_S \text{ if } k = -1 \end{array} \right\})] &= 0 \quad \forall i \geq 0, k, \\
 s_k[v_S - \sum_l s_l(\Pi_{kl} + \delta \left\{ \begin{array}{l} v_S \text{ if } k = -1, l = -1 \\ v_{Rl} \text{ if } k = -1, l \geq 0 \\ w_{Rk} \text{ if } k \geq 0, l = -1 \\ w_{Skl} \text{ if } k \geq 0, l \geq 0 \end{array} \right\})] &= 0 \quad \forall k, \\
 v_{Ri} - (\Pi_{ki} + \delta \left\{ \begin{array}{l} w_{Rk} \text{ if } k \geq 0 \\ v_S \text{ if } k = -1 \end{array} \right\}) &\geq 0 \quad \forall i \geq 0, k, \\
 v_S - \sum_l s_l(\Pi_{kl} + \delta \left\{ \begin{array}{l} v_S \text{ if } k = -1, l = -1 \\ v_{Rl} \text{ if } k = -1, l \geq 0 \\ w_{Rk} \text{ if } k \geq 0, l = -1 \\ w_{Skl} \text{ if } k \geq 0, l \geq 0 \end{array} \right\}) &\geq 0 \quad \forall k, \\
 r_{ik}, s_k &\geq 0 \quad \forall i \geq 0, k, \\
 \sum_k r_{ik} &= 1 \quad \forall i \geq 0 \text{ and } \sum_k s_k = 1, \\
 w_{Ri} &= \sum_l r_{il}(\Pi_{il} + \delta \left\{ \begin{array}{l} v_{Rl} \text{ if } l \geq 0 \\ v_S \text{ if } l = -1 \end{array} \right\}) \quad \forall i \geq 0, \\
 w_{Sij} &= \Pi_{ij} + \delta v_S \quad \forall i \geq 0, j \geq 0.
 \end{aligned}$$

We attempted to solve the problem directly as a nonlinear program using LOQO, adopting the pairwise formulation. The algorithm only

converged for a few starting points near an equilibrium. Otherwise it failed for virtually all starting points. We also tried the summation formulation; the results were similar. With either formulation, LOQO usually reaches its iteration limit (500), and the dual objective function appears to become unbounded (more on this later). To see whether this was a difficulty particular to LOQO, we also tried the problem on some other standard nonlinear solvers: KNITRO, FILTER, and SNOPT. KNITRO is an interior-point code (though still very different from LOQO in its implementation), and FILTER and SNOPT are SQP codes (see [9],[10] and [4] for detailed descriptions of these solvers). Here, too, the results were not satisfactory. All the algorithms failed for a variety of starting points (see the appendix for the exact results).

Thus it appears that this problem poses a genuine challenge for solvers when given as a nonlinear program. The question is: Why? The dual objective function reaching very high levels in LOQO is due to unbounded duals going to ∞ . However, a very large dual objective function is also often a sign of an infeasible problem. That is, when LOQO has difficulty getting feasible from a given starting point (because the constraints are locally infeasible in its neighborhood), the dual variables often get very large. So how can we distinguish between these two possible causes? This has always been a difficult question to answer (see [8]). But in this case we have some evidence that unboundedness of the multipliers is indeed a factor.

Typically, when LOQO has trouble getting feasible simply because the constraints of a problem are difficult to satisfy, the algorithm hardly moves from the starting point. On the other hand, if unbounded duals are the source of all the difficulties, some progress might be possible before the step lengths get too small. So the following result is suggestive:

For the starting point which sets all the payoffs to 63 and all the probabilities to 2 (with starting slacks at 8.1), LOQO, as usual, runs out of iterations and the dual objective increases without bound. After 500 iterations the point LOQO has reached is

$$s = (0 \ .99999 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0),$$

$$R = \begin{bmatrix} .31 & 0 & 0 & 0 & 3.00e^{-5} & .66 & .03 & 0 \\ .00077 & .996 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .44 & .56 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .996 & .001 & .002 & .002 & .002 & .001 \\ 0 & 0 & .65 & .35 & 0 & 0 & 0 & 0 \\ 0 & 0 & .62 & .06 & .32 & 0 & 0 & 0 \\ 0 & 0 & .62 & .06 & .32 & 0 & 0 & 0 \end{bmatrix}.$$

From [18] and [19] it is clear that this point appears to be agonizingly close to an equilibrium, indicating that unbounded multipliers may have prematurely stalled progress of the algorithm. The following example is also interesting:

We experimented with a very simple, but related model. In this model, the firms only move simultaneously, and each time they must choose a price (no option to wait). All other rules remain the same. The equations for the MPEC are:

$$s_k[v_S - \sum_l s_l(\Pi_{kl} + \delta v_S)] = 0 \quad \forall \text{ prices } k,$$

$$v_S - \sum_l s_l(\Pi_{kl} + \delta v_S) \geq 0, s_k \geq 0 \quad \forall \text{ prices } k,$$

$$\sum_k s_k = 1.$$

This model has exactly two equilibria, if we keep the same payoff structure and discount factor as before: both firms set the price to 1 with probability 1, or to 0 with probability 1 (and associated payoffs are 25 and 0 respectively). Even on this problem LOQO fails from a few starting points (though not from as many as for the full model). One such starting point sets the payoff variable to 70 and all probabilities to .14 (and starting slacks set to .0001). With this starting point, as before LOQO runs out of iterations and the dual objective function grows without bound. After 500 iterations the point LOQO has reached is

$$\text{payoff} = 23.03, s = (.09 \ .52 \ -1.77e^{-17} \ .03 \ .13 \ .09 \ .1).$$

It is immediately clear that LOQO was moving towards one of the equilibria (price = 1, payoff = 25), when it stalled. This is even more interesting than the previous example. LOQO can sometimes have numerical difficulties near a solution. In this case it stalled near neither a solution nor the starting point, but somewhere in between, while

apparently making good progress. This argues still more strongly in favor of unbounded duals causing difficulties (though this is not to say that there aren't other complicating factors involved, which are demonstrated below).

We also looked at the simpler model which is analysed in [17], [18] and [19]. In this model the firms always alternate their moves, and here too there is no option to wait. Thus each round one firm is moving, and the other is committed to its price from the previous round. We ran this model on LOQO, and failure was still common for many starting points. This time, the summation formulation was clearly better than the pairwise, though it too still failed for various starting points (the pairwise failed almost everywhere). This is not surprising behavior, as the summation formulation can indeed be better in the presence of unbounded multipliers, as then there are fewer multipliers that can go to ∞ (and hence fewer slack variables which become small).

5.1. Solving the problem using penalty methods. The ℓ_1 and ℓ_∞ penalty methods worked very well on all three models discussed (the main model, as well as the alternating-move and simultaneous-move models). Both methods work for almost all starting points, including starting slack values. They fail on only two (rare) occasions:

1. If the starting slack value is very small (as mentioned before, LOQO can stall if slack variables get too close to 0).
2. From some starting points, the algorithm returns what might be called a "false" solution, i.e. a solution to the penalized problem which is infeasible for the true problem. Meaning in this case, all the constraints are satisfied except that the complementarity is positive. We knew this could happen, since these methods are not globally convergent. Still, this happens rarely, and changing the starting point just slightly generally fixes the problem.

Here are two examples of the equilibria we found using the penalty method:

Setting starting probabilities to .2 and starting payoffs to 30, LOQO reached the following equilibrium after 57 iterations:

$$s = (0 \ .83878 \ 0 \ 0 \ 0 \ 0 \ 0 \ .16122),$$

$$R = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & .619308 & .380692 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Setting starting probabilities to .8 and payoffs to 63, LOQO reached the following equilibrium after 60 iterations:

$$s = (0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0),$$

$$R = \begin{bmatrix} .360441 & 0 & 0 & 0 & 0 & .152644 & .379981 & .106935 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

To try to improve convergence still further, we tried to modify the penalty problem (10) by adding an upper bound to the infeasibility of the complementarity constraint, ζ , and gradually tightening it as needed. For a few starting points from which we had been getting to a "false" optimum before, now we got to a true solution. But performance was actually worse for most starting points. In fact, setting the bound on ζ to .05 resulted in failure from almost all starting points. In practice it appears best to leave the infeasibility unbounded above, and shift the starting point, if necessary, to get a true solution to the problem.

The findings just described are crucial. We have an example of a problem with bounded multipliers at the solutions (and which hence obeys constraint qualifications) on which, nonetheless, LOQO fails from almost everywhere. The only explanation for this is the extreme non-convexity of the problem, which is due to the problem having a large set of discrete equilibria. The discreteness arises from the strict complementarity of the solutions, which is typical for a game theoretic problem, as at equilibrium actions are rarely played with probability 0 if they are as good as the actions actually being played. Such a non-convex problem can have many locally-infeasible points.

Definition 1. A point x is locally infeasible if $h_i(x) \neq 0$ for some i , and x is a local minimizer of the function $\sum_i |h_i(x)|$ (assuming a problem with feasible set $h_i(x) = 0, i = 1, \dots, m$).

Algorithms commonly converge to locally infeasible points, and to confirm that many locally feasible points were present, we also attempted to solve the problem by relaxing *all* the constraints (i.e. not just the complementarity constraints). Indeed, from most starting points we got non-zero solutions. This suggests that unbounded multipliers are not the only factor, or perhaps even the main factor, making our problem difficult. This is not very surprising, as unbounded multipliers can be expected to cause trouble only if a problem is already difficult for some reason. We know that penalty methods bound the multipliers, and that they can remove this extreme non-convexity may be explained as follows:

Our problem has the form

$$(13) \quad \begin{array}{ll} \min & 0 \\ \text{s.t.} & x_i g_i(x) = 0, \quad i = 1, \dots, n, \\ & x \geq 0, g(x) \geq 0, \\ & h_j(x) = 0, \quad j = 1, \dots, m. \end{array}$$

If we consider the ℓ_∞ method, the penalized problem in this case looks like:

$$(14) \quad \begin{array}{ll} \min & \zeta \\ \text{s.t.} & x_i g_i(x) \leq \zeta, \quad i = 1, \dots, n, \\ & \zeta \geq 0, x \geq 0, g(x) \geq 0, \\ & h_j(x) = 0, \quad j = 1, \dots, m. \end{array}$$

Assuming g and h are well behaved, we expect (14) to have a continuous feasible set, indeed to obey standard constraint qualifications. In other words, (14) differs from (13) in that it removes the complicating constraint from (13). Thus the constraints in (14) can be expected to produce far fewer locally infeasible points than those in (13). (14) can of course produce "false" optima as described above, but these can only occur over the feasible set. That is, the region in which the algorithm can fail to converge to a true solution is much smaller for (14) than (13). (14) also bounds the multipliers, of course, so we believe this is a comprehensive explanation for why the penalty method is greatly superior to the direct approach on this problem. As there can be many kinds of equilibrium problems which produce large, discrete feasible sets (game theory being only one example), this indicates that penalty methods could be indispensable for solving MPEC's in general.

6. OTHER MPEC'S

We also tested the efficacy of penalty methods on the MacMPEC test set found in ([21]). As these problems are mostly true MPEC's (i.e. with an objective function), two major issues were how to choose the initial value of the penalty parameter, and how to update it during the course of the algorithm. Using LOQO, we chose to implement the ℓ_∞ penalty method to relax the complementarity constraints. The first step was to initialize the penalty parameter.

Initialization was done as follows: Given an initial value λ_o for the Lagrange multiplier of the bound constraint on the auxiliary variable ζ , and initial values $\lambda_1, \dots, \lambda_m$ for all the other constraints, we set the initial value of the parameter ρ to be simply

$$\rho = \lambda_o + \sum_{i=1}^m \lambda_i.$$

The initial Lagrange multiplier estimates are obtained by solving the system

$$\begin{bmatrix} -H + I & A^T \\ A & I \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} \nabla f(x^0) \\ h(x^0) \end{bmatrix}$$

where H, A, h are as defined in Section 2. The system above is derived from the KKT system shown in Section 2, with θ set to 0. We emphasize that all the multiplier estimates (not just those for the complementarity constraints) are used in the calculation of the initial ρ . This is done to get a better estimate of the order of magnitude of the numbers in the problem.

The perturbation variable is initialized at the value of the LOQO option `bndpush`, which is also the initial value of all the slack variables. The default value of `bndpush` is 1.

The updating of the penalty parameter occurs whenever the algorithm is approaching optimality with a scaled perturbation, $\hat{\zeta}$, that is greater than 10^{-6} . The scaling of the perturbation is done as follows:

$$\hat{\zeta} = \frac{\zeta}{\max_i(1, x_i, g_i(x))}$$

The scaled perturbation is used as a better measure of what constitutes "0" given the magnitude of the numbers in the problem.

When $\hat{\zeta} > 10^{-6}$ at the conclusion of the algorithm, the penalty parameter and the dual variable associated with the bound on the perturbation are updated:

$$\rho^{k+1} = 10\rho^k, \quad \lambda_o^{k+1} = \lambda_o^k + \rho^{k+1} - \rho^k$$

After the update, the algorithm continues to make progress from the previous values for all other variables, thereby performing a "warm start". Updating the dual variable as given ensures that dual feasibility will be maintained.

Note that at present we do not have a way to deal with the situation where LOQO fails to find a solution at all (i.e. even a "false" optimum) for too small a value of the penalty parameter. An example of this behavior is mentioned in the numerical results below. It will require further investigation.

6.1. Numerical Results. We tried our algorithm on two test-sets: the MacMPEC test-set already mentioned (found at [21]), and a test-set, described in [5], of pure complementarity problems which have usually been regarded as difficult for complementarity solvers. We were able to solve almost all the problems successfully. Moreover, our technique for estimating the initial value of the penalty parameter proved quite effective. The penalty method was able to solve some problems which a standard implementation of LOQO was unable to solve. These problems were *ex9.2.2*, *pack-rig2-16*, *pack-rig2p-32*, *qpec2*, *ralph1*, *scholtes4*, *tap-09*, and *tap-15*.

The full results are given in the appendix. We give not only the results for LOQO, but for sake of comparison, the results for a standard SQP solver, FILTER (see [9] for details on FILTER). As reported in [8], FILTER also appears to be quite successful, particularly on the MacMPEC set. Now we mention those few problems which did cause some difficulties for LOQO. There were seven such problems in the MacMPEC set, labeled as *ex9.2.2*, *ralph1*, *scholtes4*, *ralph2*, *siouxfls1*, *water-net* and *water-FL*.

ralph2 is the problem

$$\begin{array}{ll} \min & x^2 - y^2 - 4xy \\ \text{s.t.} & xy = 0, \\ & x \geq 0, y \geq 0. \end{array}$$

The ℓ_∞ penalty method applied to this problem results in the penalized problem

$$\begin{array}{ll} \min & x^2 - y^2 - 4xy + \rho\zeta \\ \text{s.t.} & xy \leq \zeta, \\ & x \geq 0, y \geq 0, \end{array}$$

which may be rewritten as

$$\begin{array}{ll} \min & (x - y)^2 - 2xy + \rho\zeta \\ \text{s.t.} & xy \leq \zeta, \\ & x \geq 0, y \geq 0. \end{array}$$

It can be seen by inspection that for $\rho < 2$, making x and y arbitrarily large while keeping $x = y$, and keeping $\zeta = xy$, makes the objective function arbitrarily small while staying within the feasible set. In other words, the penalized problem is unbounded for too small a value of the penalty parameter. As a result, LOQO might not return a solution, and our method as outlined would stall. In this case increasing ρ to be at least 2 resolves the issue. But as yet we have not found a way to handle the situation in general in which too small a penalty might produce an unbounded (or otherwise badly-behaved) problem, not to mention cases where the penalized problem can remain badly-behaved for any finite value of the penalty. This is all the subject of future research.

ralph1 is the problem

$$\begin{array}{ll}\min & 2x - y \\ \text{s.t.} & y(y - x) = 0, \\ & y \geq 0, y - x \geq 0, x \geq 0.\end{array}$$

The KKT conditions are

$$\begin{aligned}2 - \lambda y + \mu_1 - \mu_3 &= 0, \\ -1 + 2\lambda y - \lambda x - \mu_1 - \mu_2 &= 0,\end{aligned}$$

(plus the complementarity and feasibility conditions), where λ is the multiplier for the complementarity constraint, and μ_1 corresponds to $y - x \geq 0$, μ_2 to $y \geq 0$, and μ_3 to $x \geq 0$.

It is clear by inspection that the point $(0, 0)$ is the unique solution to this problem. But substituting $(0, 0)$ for (x, y) in the second equation above yields

$$-1 = \mu_1 + \mu_2$$

where μ_1 and μ_2 are nonnegative multipliers as they correspond to inequality constraints. Thus the KKT system has no solution at the point $(0, 0)$.

The penalized version of the above problem is

$$\begin{array}{ll}\min & 2x - y + \rho\zeta \\ \text{s.t.} & y(y - x) \leq \zeta, \\ & \zeta \geq 0, y \geq 0, y - x \geq 0, x \geq 0.\end{array}$$

The KKT conditions are

$$\begin{aligned}2 - \lambda y + \mu_1 - \mu_3 &= 0, \\ -1 + 2\lambda y - \lambda x - \mu_1 - \mu_2 &= 0, \\ \rho &= \lambda + \mu_4,\end{aligned}$$

(plus complementarity and feasibility conditions).

Again it is the case that at the point $(0, 0)$ there is no nonnegative solution for (μ_1, μ_2) , for any value of the penalty. Hence any nonlinear solver which tries to find KKT points will run into trouble, and as noted in the appendix, LOQO could not find the solution to this problem. Interestingly, what it did find was the point $(0, \frac{1}{2\rho})$, with $\zeta = \frac{1}{4\rho^2}$, for a fixed ρ , which does admit a KKT solution. As ρ gets large this solution will asymptotically converge to the non-KKT point $(0, 0)$. In other words, the usually exact ℓ_∞ penalty function starts to resemble an inexact function (exactness of course cannot hold when no KKT solution exists). There is reason to think that this kind of behavior may be observed in general:

For a penalized problem

$$\min f(x) + \rho P(x)$$

where $P(x) = 0$ represents a feasible point, as ρ gets very large, solutions to the above problem would be expected to converge to a point which locally minimizes $P(x)$. Thus for a good enough starting point convergence to the solution to the original problem can be achieved, even if that solution is not a KKT point. Moreover, this would apply to problems in general with non-KKT points as solutions, not just MPEC's. In fact, we have observed the same behavior for problem 13 of the Hock and Schittkowski test set [11]. We now prove this quickly under fairly general conditions.

Theorem 2. *Given the following nonlinear programming problem:*

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & h_i(x) \geq 0, i = 1, \dots, m. \end{aligned}$$

Let $f(x)$ and $h_i(x)$ for all i be twice continuously differentiable. Let x^ be a solution to the above problem. Suppose that x^* is the unique solution in a neighborhood of $B_r(x^*)$, $r > 0$. Let there be given a monotonic sequence $0 < \rho_1 < \dots < \rho_k$ such that $\lim_{k \rightarrow \infty} \rho_k = \infty$ and let (x_k, ζ_k) be the unique solution with $x \in B_r(x^*)$ to the problem*

$$(15) \quad \begin{aligned} \min \quad & f(x) + \rho_k \zeta \\ \text{s.t.} \quad & h_i(x) \geq -\zeta, i = 1, \dots, m, \\ & \zeta \geq 0. \end{aligned}$$

Then $\lim_{k \rightarrow \infty} x_k = x^$.*

Proof. By assumption, we have for all k ,

$$\begin{aligned} f(x_k) + \rho_k \zeta_k &\leq f(x_{k+1}) + \rho_k \zeta_{k+1}, \\ f(x_{k+1}) + \rho_{k+1} \zeta_{k+1} &\leq f(x_k) + \rho_{k+1} \zeta_k. \end{aligned}$$

Therefore,

$$(\rho_{k+1} - \rho_k)\zeta_k \geq (\rho_{k+1} - \rho_k)\zeta_{k+1} \quad \forall k,$$

which implies that

$$\zeta_{k+1} \leq \zeta_k \quad \forall k.$$

In other words, $\{\zeta_k\}$ is a monotonically decreasing sequence bounded from below, hence it converges to some $\hat{\zeta}$. The first inequality above also gives us

$$f(x_{k+1}) - f(x_k) \geq \rho_k(\zeta_k - \zeta_{k+1}) \geq 0 \quad \forall k.$$

Thus $\{f(x_k)\}$ is an increasing sequence. Since $(x^*, 0)$ is feasible for (15), for any ρ_k , we have

$$(16) \quad f(x_k) + \rho_k \zeta_k \leq f(x^*) \quad \forall k,$$

which implies $\{f(x_k)\}$ is bounded above. Hence $\{f(x_k)\}$ converges to some $\{f(\hat{x})\}$, and by the continuity of f , $\{x_k\}$ converges to \hat{x} . We now only need to show that $\hat{\zeta} = 0$:

(16) implies

$$\zeta_k \leq \frac{f(x^*) - f(x_k)}{\rho_k} \quad \forall k.$$

This gives

$$\hat{\zeta} = \lim_{k \rightarrow \infty} \zeta_k \leq \lim_{k \rightarrow \infty} \frac{f(x^*) - f(x_k)}{\rho_k} \leq \lim_{k \rightarrow \infty} \frac{f(x^*) - f(\hat{x})}{\rho_k} = 0.$$

Since $\hat{\zeta} = 0$, it follows that \hat{x} must be feasible. Since $\{f(x_k)\}$ is bounded above by $f(x^*)$, we have $f(\hat{x}) \leq f(x^*)$. The last equality follows from the fact that $\rho_k \rightarrow \infty$ as $k \rightarrow \infty$. Therefore $\hat{x} = x^*$, since x^* is by assumption the only solution to our problem. \square

The crucial point to note is that no assumption is made about constraint qualifications. All that is required for this theorem to be applicable in some neighborhood of a solution is that the sequence of penalized problems have KKT solutions. This can be ensured if the constraint gradient matrix for the penalized problems is nonsingular, and this can be guaranteed for the ℓ_1 penalty function (see the next Section).

The problems *ex9.2.2* and *scholtes4*, like *ralph1* also do not have KKT points and similarly allow for only approximate answers. We note that these three problems have non-strict solutions (the extra linear dependence is what keeps multipliers from existing), but the non-strictness as such did not cause any problems for the penalty method. Indeed, there are over 50 problems in the MacMPEC set which have at least one non-strict pair, and with the exception of the ones just

described, the penalty method solved them all. This is in contrast to what happens when LOQO is used to solve the problems directly, without relaxation. In that case three non-strict problems (*tap-09*, *tap-15*, and *qpec2*), which do have KKT solutions, caused difficulties. These are examples of rank-deficiency (which occurs in non-strict problems, as shown in Section 3) hurting an interior-point method for numerical reasons. The penalty method avoids the issue of rank deficiency altogether by keeping all multipliers within the system (and bounded). The problems *siouxfls1*, *water-FL* and *water-net* are rank-deficient even after relaxation, i.e. the non-complementarity constraints are not well-behaved. Hence trouble with these problems is not surprising.

The other test set (of pure complementarity problems) consisted of 11 models. Of these we solved 8, with minor adjustments as needed (described in the appendix). One model had serious errors which caused the AMPL processor to fail, and the two problems on which genuine failure occurred had functions which were not differentiable at some point. This can cause serious numerical difficulties for any method based on the assumption of differentiability.

Thus the numerical evidence shows that penalty methods, in conjunction with an interior-point solver, can successfully resolve the difficulties posed by equilibrium constraints. In one framework we are able to deal with strict and non-strict problems, and even problems without KKT solutions. Moreover our implementation does not even require that the multipliers at the solution be unique, merely bounded. This is due, we surmise, to the property that interior-point methods have of going to the center of the face of solutions. And we have some numerical evidence for this. Consider, for example, the problem *tap-15*: the optimal value of the penalty parameter was 1000. There were 83 complementarity constraints, and their optimal multipliers summed to 994, with all of them having values between 9 and 12, and most of them being almost equal. In other words, the algorithm really did go to the center of the face of optimal solutions.

7. PENALTY METHODS FOR NONLINEAR PROGRAMMING

In the previous section, we showed that penalty methods can aid in finding the optimal solution when no KKT point exists. It can be seen from the proof of Theorem 3 that this property applies in general to NLPs. In this section, we outline two more issues that can be resolved using penalty methods in the context of general NLP.

7.1. Jamming in Interior-Point Methods. A problem that interior-point methods such as LOQO have experienced in the past is that of

jamming. Jamming occurs when variables that must be kept non-negative (in LOQO's case, the slack variables) attain a value of 0 and the descent direction followed by the algorithm in the next iteration would make them negative. In this situation an interior-point method is forced to cut its steplength to 0 and the algorithm stalls. In [3] it is shown that jamming will not occur in LOQO for non-degenerate problems (i.e. problems with strict complementarity throughout between slack and dual variables) if the constraint gradient matrix for a particular subset of the constraints is nonsingular. We now show that the use of the ℓ_1 penalty method to relax all the constraints of a problem will ensure that this is always the case.

(In what follows, once again as is standard with the notation of interior-point methods, a capitalized variable name represents a diagonal matrix whose elements are the coordinates of the corresponding vector variable; \mathcal{Z} will be used for a capital ζ .)

For simplicity of notation we will consider problems of the form

$$(17) \quad \begin{array}{ll} \min & f(x) \\ \text{s.t.} & h_i(x) \geq 0, \quad i = 1, \dots, m. \end{array}$$

Theorem 3. *Assuming (17) is non-degenerate as defined above, LOQO will not jam in the sense of [3] if the ℓ_1 penalty method is applied.*

Proof. We relax each constraint with a separate auxiliary variable to obtain the penalized problem

$$(18) \quad \begin{array}{ll} \min & f(x) + \rho \sum_{i=1}^m \zeta_i \\ \text{s.t.} & \zeta_i \geq 0, g_i(x) \geq -\zeta_i, \quad i = 1, \dots, m. \end{array}$$

The auxiliary variables are then treated like slack variables and put in the barrier function:

$$(19) \quad \begin{array}{ll} \min & f(x) + \rho \sum_{i=1}^m \zeta_i - \theta_1 \sum_{i=1}^m \log \zeta_i \\ \text{s.t.} & g_i(x) + \zeta_i \geq 0, \quad i = 1, \dots, m. \end{array}$$

The constraint gradient matrix of (19) is

$$\begin{bmatrix} & & I & \\ \nabla g_1(x) & . & . & \nabla g_m(x) \end{bmatrix}$$

which will always be nonsingular.

We then add a slack variable to each of the original constraints and put them in the barrier function as usual:

$$(20) \quad \begin{aligned} \min \quad & f(x) + \rho \sum_{i=1}^m \zeta_i - \theta_1 \sum_{i=1}^m \log \zeta_i - \theta_2 \sum_{i=1}^m \log w_i \\ \text{s.t.} \quad & g_i(x) - w_i + \zeta_i = 0, \quad i = 1, \dots, m. \end{aligned}$$

The Lagrangian for the above problem is

$$\mathcal{L}(x, w, \zeta, \lambda) = f(x) + \rho \sum_{i=1}^m \zeta_i - \theta_1 \sum_{i=1}^m \log \zeta_i - \theta_2 \sum_{i=1}^m \log w_i - \lambda^T (g(x) - w + \zeta).$$

The KKT conditions are

$$\begin{aligned} \nabla f(x) - g(x)^T \lambda &= 0, \\ \rho e - \theta_1 Z^{-1} e - \lambda &= 0, \\ -\theta_2 W^{-1} e + \lambda &= 0, \\ g(x) - w + \zeta &= 0. \end{aligned}$$

The second equation of the above system may be rewritten as

$$\rho Z e - Z \lambda = \theta_1 e,$$

which in turn is equivalent to

$$(\rho I - \Lambda) Z e = \theta_1 e \text{ (assuming } \zeta > 0 \text{)}.$$

Similarly the third equation is equivalent to

$$W \Lambda e = \theta_2 e.$$

We note that the second condition implies that the auxiliary variables ζ be nonnegative at all times, and that the boundedness of the dual variables λ guarantees that there exists a finite ρ such that $(\rho I - \Lambda)$ is strictly positive. In that case, the system above will converge to the true system (for (17)) as $\theta_1, \theta_2 \rightarrow 0$ if $\zeta \rightarrow 0$ as well. And, *Newton's method applied to this system will not jam*. This conclusion is valid because the KKT system for (20) is the system that is obtained for the barrier problem applied to (19) for any particular value of θ_1 . (19) is an optimization problem with, as already noted, a nonsingular constraint gradient matrix at all times. This along with the nondegeneracy assumption meets the requirements of the anti-jamming theorem in [3]. Finally, it can be seen that the nonnegative variables ζ will not "jam" either against their boundaries:

The Newton system for the KKT system to (20) will produce the equation

$$(\rho I - \Lambda) \Delta \zeta - \zeta \Delta \lambda = \theta_1 e,$$

which implies that if $\zeta = 0$, $\Delta\zeta > 0$. \square

Note that the above conclusion will not hold if the ℓ_∞ penalty method is used. This is an important theoretical distinction between the two exact penalty methods.

7.2. Infeasibility Detection. A further use of penalty methods is in infeasibility detection, meaning the identification of a point as locally infeasible. For this purpose, of course, all constraints in a problem need to be relaxed. We did so for one of the problems in the MacMPEC test set which is known to be infeasible, *pack-rig2c-16*, using the ℓ_∞ penalty function. We obtained the following results:

penalty	optimal perturbation
10	1.102629E-01
100	1.270534E-01
1000	1.026225E-01
10000	1.026297E-01

Not unexpectedly, the algorithm reaches a point where it is unable to improve feasibility any further, regardless of how large it makes the penalty. Thus relaxing all constraints provides a way to identify local infeasibility as the cause for an algorithm to stall. And we plan to implement this as an “elastic mode” for LOQO in the future, to do exactly that.

8. CONCLUSION

In this paper we discussed the question of solving an MPEC as a nonlinear program. This poses theoretical difficulties, both in terms of unbounded Lagrange multipliers at a solution, and often, a discrete feasible set resulting in extreme non-convexity. We gave an example where these difficulties have a real effect and showed that penalty methods can resolve these difficulties in theory, and do so in practice for our example. We went on to implement a version of the penalty method with the interior-point code LOQO, which successfully solved almost all the problems in two standard test-sets. Furthermore, our penalty method solved some problems elegantly which before had required an ad-hoc approach. The clear conclusion is that penalty methods appear to be a good way to solve MPEC's and equilibrium problems in general using nonlinear solvers. We have even shown that the ℓ_1 penalty method can actually resolve a general difficulty that interior-point methods such as LOQO have experienced in the past. Thus the utility of penalty methods potentially extends far beyond equilibrium problems.

With regards to equilibrium problems, a robust tool for computing equilibria can render difficult but important questions tractable, and open up a vast number of areas for application. In particular, the stability of equilibria and the dynamics of systems in the vicinity of equilibria can be studied more successfully if the equilibria can actually be found. In terms of applications, game theory is both a source of some of the hardest equilibrium problems (such as the one presented here), and a subject of intense scrutiny at present because of its applicability to many important areas. The energy sector is one such area, while another is auctions. Auctions are becoming increasingly important, both in public sector activities, e.g. spectrum auctions for wireless companies, and in the private sector, for instance in business-to-business e-commerce. Thus there should be an abundance of important, realistic problems to consider.

8.1. Acknowledgement. We would like to thank Sven Leyffer for providing us with the FILTER code for use in comparative testing.

REFERENCES

- [1] Anitescu, Mihai. "Nonlinear Programs with Unbounded Lagrange Multiplier Sets." Technical Report ANL/MCS-P793-0200.
- [2] Benson, Hande Y. and David F. Shanno and Robert J. Vanderbei. "Interior Point Methods for Nonconvex Nonlinear Programming: Filter Methods and Merit Functions." *Computational Optimization and Applications* 23, 2002, 257-272.
- [3] Benson, Hande Y. and David F. Shanno and Robert J. Vanderbei. "Interior Point Methods for Nonconvex Nonlinear Programming: Jamming and Comparative Numerical Testing." Princeton University ORFE Technical Report 00-02. *Math. Programming* (in press).
- [4] Byrd, R.H. and M. E. Hribar and J. Nocedal. "An Interior Point Algorithm for Large Scale Nonlinear Programming." *SIAM Journal on Optimization* 9(4):877-900, Sept. 1999.
- [5] Ferris, Michael C. and S. P. Dirkse and A. Meeraus. "Mathematical Programs with Equilibrium Constraints: Automatic Reformulation and Solution via Constrained Optimization." Oxford University Numerical Analysis Group Research Report NA-02/11.
- [6] Ferris, Michael C. and Christian Kanzow. "Complementarity and Related Problems: A Survey." University of Wisconsin Mathematical Programming Technical Report 98-17, November 1998.
- [7] Ferris, Michael C. and Jong-Shi Pang. "Engineering and Economics Applications of Complementarity Problems." *SIAM Review* 39 (1997) pp 669-713.
- [8] Fletcher, R. and S. Leyffer. "Numerical Experience with Solving MPEC's as NLP's." University of Dundee Technical Report NA-210, August 2002.
- [9] Fletcher, R. and S. Leyffer and Philippe L. Toint. "On the Global Convergence of a FILTER-SQP Algorithm." University of Dundee Report NA197.
- [10] Gill, Philip E. and W. Murray and M. A. Saunders. "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization." *SIAM Journal on Optimization* 12 (2002), 979-1006.
- [11] Hock, W. and Schittkowski, K. *Test examples for nonlinear programming codes*. Lecture Notes in Economics and Mathematical Systems 187. Springer Verlag, Heidelberg, 1981.
- [12] Huang, Jacqueline and Jong-Shi Pang. "Option Pricing and Linear Complementarity." *Journal of Computational Finance*, vol 2 no. 3, 1998.
- [13] Leyffer, S. MacMPEC Test Suite. <http://www-unix.mcs.anl.gov/~leyffer/MacMPEC>
- [14] Scheel, H and S. Scholtes. "Mathematical Program with Complementarity Constraints: Stationarity, Optimality and Sensitivity." *Mathematics of Operations Research* 25, 1-22, 2000.
- [15] Scholtes, Stefan. "Convergence Properties of a Regularization Scheme for Mathematical Programs with Complementarity Constraints." *SIAM Journal on Optimization* 11, 918-936, 2001.
- [16] Sen, A. Duopoly model. <http://www.princeton.edu/~asen>.
- [17] Simantiraki, Evangelia M. and David F. Shanno. "An Infeasible Interior-Point Algorithm for Solving Mixed Complementarity Problems." Rutgers University RUTCOR Research Report 37-95.

- [18] Tirole, Jean and Eric Maskin. "A Theory of Dynamic Oligopoly I: Overview with Quantity Competition and Large Fixed Costs." *Econometrica* 1988.
- [19] Tirole, Jean and Eric Maskin. "A Theory of Dynamic Oligopoly II: Price Competition." *Econometrica*, 1988.
- [20] Vanderbei, Robert J. and David F. Shanno. "An Interior-Point Algorithm for Nonconvex Nonlinear Programming." *Computational Optimization and Applications* 13, 1999, 231-252.
- [21] Waechter, Andreas. "An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering", Ph.D. thesis. January 2002.

APPENDIX A. PERFORMANCE OF OTHER NONLINEAR CODES ON THE MAIN (GAME THEORY) MODEL

We tested the main model on some of the nonlinear codes on the NEOS server (www-neos.mcs.anl.gov/neos/). A naive type of starting point was used in which all the probabilities are set to the same number (between 0 and 1) and all the payoff variables are set to the same number. The results were as follows:

PAIRWISE FORMULATION			
start pt.	solver	result	reason given (if any)
.8/63	SNOPT	fail	infeasible problem
.8/63	FILTER	optimal	
.8/63	KNITRO	optimal	
.8/80	SNOPT	fail	no further improvement possible
.8/80	FILTER	fail	locally infeasible
.8/80	KNITRO	optimal	
.2/20	SNOPT	optimal	
.2/20	FILTER	fail	locally infeasible
.2/20	KNITRO	fail	iteration limit

SUMMATION FORMULATION			
start pt.	solver	result	reason given (if any)
.8/63	SNOPT	fail	infeasible problem
.8/63	FILTER	fail	locally infeasible
.8/63	KNITRO	fail	iteration limit
.2/20	SNOPT	optimal	
.2/20	FILTER	optimal	
.2/20	KNITRO	optimal	

APPENDIX B. PERFORMANCE OF PENALTY METHODS ON OTHER MPEC'S

For the numerical testing, the two solvers used were: LOQO (Version 6.04) (see [20] for details) and filterMPEC (Version 20020621) (see [9]). The test problems come from the MacMPEC test suite by Leyffer ([21]) and the set described in [5] by Ferris, and all were formulated in AMPL. All testing was conducted on a Dell Workstation running Red Hat Linux 9 with 5GB of main memory and 864MHz clock speed. Results of comparative testing are provided in the tables below. This information is intended to display the relative robustness of the codes. Runtimes are not included in the comparison because of two reasons: (1) Most of the problems are solved in under 0.01 CPU seconds by both solvers, and (2) the solvers were compiled using different compilers (LOQO with GNU compiler gcc, and FILTER using Intel compilers) which can lead to significant differences in runtimes. The first set of tables pertain only to LOQO's performance on the MacMPEC set, as some of the information displayed there is not relevant for the comparison to FILTER. The next set of tables compare LOQO and FILTER on the MacMPEC set, and the last group of tables show the results on the Ferris set for both solvers.

MacMPEC testset LOQO results								
problem	vars	cons	iters	time	optimal obj. fn.	final relax	initial pen	final pen
bard1	9	7	13	0	1.700000E+01	1.38E-09	1.00E+01	1.00E+01
bard1m	10	7	13	0	1.700000E+01	2.26E-09	1.00E+01	1.00E+01
bard2	16	12	36	0	6.598000E+03	3.46E-11	1.00E+03	1.00E+03
bard2m	16	12	21	0	-6.598000E+03	5.43E-10	1.00E+03	1.00E+03
bard3	8	6	19	0	-1.267871E+01	5.18E-10	1.00E+01	1.00E+01
bard3m	10	8	24	0	-1.267871E+01	3.99E-10	1.00E+01	1.00E+01
bar-truss-3 ^f	42	40	29	.01	1.016657E+04	2.51E-13	1.00E+04	1.00E+04
bem-milanc30-s	4901	4897	127	59.2	9.443495E-02	1.30E-11	1.00E+03	1.00E+03
bilev1	13	13	27	0	1.117272E-09	1.19E-13	1.00E+02	1.00E+02
bilev2	25	21	22	.01	-6.600000E+03	2.79E-10	1.00E+03	1.00E+03
bilevel1	17	15	37	.01	3.851617E-09	1.08E-12	1.00E+02	1.00E+02
bilevel2	25	21	22	.01	-6.600000E+03	2.97E-10	1.00E+03	1.00E+03
bilevel3	15	13	24	0	-1.267871E+01	1.03E-11	1.00E+02	1.00E+02
bilin	15	13	31	0	1.460000E+01	1.85E-12	1.00E+02	1.00E+02
dempe	5	3	15	0	3.125001E+01	2.93E-11	1.00E+01	1.00E+01
design-cent-1	16	14	25	0	1.860647E+00	1.37E-11	1.00E+01	1.00E+01
design-cent-2	17	18	24	.01	3.483816E+00	1.15E-10	1.00E+01	1.00E+01
design-cent-3	19	14	24	.02	3.723370E+00	7.88E-14	1.00E+02	1.00E+02
design-cent-4	31	28	26	.01	-2.130414E-09	2.22E-10	1.00E+01	1.00E+01
desilva	9	6	23	0	-1.000000E+00	2.45E-10	1.00E+00	1.00E+00
df1	4	4	24	0	3.109960E-09	1.15E-10	1.00E+01	1.00E+01
ex9.1.1	19	17	15	0	-1.300000E+01	4.98E-10	1.00E+02	1.00E+02

MacMPEC testset LOQO results								
problem	vars	cons	iters	time	optimal obj. fn.	final relax	initial pen	final pen
ex9.1.10	15	12	12	0	-3.250000E+00	1.25E-09	1.00E+01	1.00E+01
ex9.1.2	11	10	13	0	-6.250000E+00	1.32E-09	1.00E+02	1.00E+02
ex9.1.3	30	27	37	.01	-2.920000E+01	1.27E-10	1.00E+02	1.00E+02
ex9.1.4	11	9	15	0	-3.700000E+01	5.45E-10	1.00E+02	1.00E+02
ex9.1.5	19	17	13	0	-1.000000E+00	1.08E-10	1.00E+02	1.00E+02
ex9.1.6	21	19	28	.01	-4.900000E+01	5.51E-11	1.00E+02	1.00E+02
ex9.1.7	24	21	29	.01	-2.600000E+01	6.89E-11	1.00E+02	1.00E+02
ex9.1.8	15	12	12	0	-3.250000E+00	1.25E-09	1.00E+01	1.00E+01
ex9.1.9	18	16	17	0	3.111111E+00	1.16E-09	1.00E+01	1.00E+01
ex9.2.1	15	13	14	0	1.700000E+01	1.73E-10	1.00E+02	1.00E+02
ex9.2.2*†	10	9	42	0	9.999167E+01	8.33E-07	1.00E+02	1.00E+03
ex9.2.3	19	17	17	0	5.000000E+00	4.34E-11	1.00E+02	1.00E+02
ex9.2.4	11	9	11	0	5.000000E-01	2.72E-09	1.00E+01	1.00E+01
ex9.2.5	12	10	14	0	9.000000E+00	5.30E-11	1.00E+02	1.00E+02
ex9.2.6	23	18	23	0	-1.000000E+00	6.71E-10	1.00E+01	1.00E+01
ex9.2.7	15	13	14	0	1.700000E+01	1.73E-10	1.00E+02	1.00E+02
ex9.2.8	9	7	3	0	1.500000E+00	8.35E-11	1.00E+02	1.00E+02
ex9.2.9	13	11	13	0	2.000000E+00	3.54E-11	1.00E+01	1.00E+01
flp2	7	4	13	0	4.150241E-09	4.15E-10	1.00E+01	1.00E+01
flp4-1	111	90	37	.6	5.933353E-09	1.12E-11	1.00E+02	1.00E+02
flp4-2	171	170	33	1.65	7.093403E-10	1.07E-12	1.00E+02	1.00E+02
flp4-3	211	240	139	16.41	1.881357E-09	2.95E-13	1.00E+03	1.00E+03
flp4-4	301	350	65	40.08	2.386228E-08	4.61E-13	1.00E+04	1.00E+04
gauvin	6	4	18	0	2.000000E+01	1.33E-10	1.00E+01	1.00E+01
gnash10	22	20	46	.02	-2.308232E+02	9.99E-13	1.00E+03	1.00E+03
gnash11	22	20	58	.02	-1.299119E+02	5.53E-12	1.00E+03	1.00E+03
gnash12	22	20	30	.01	-3.693311E+01	4.58E-13	1.00E+03	1.00E+03
gnash13	22	20	45	.02	-7.061783E+00	9.47E-14	1.00E+03	1.00E+03
gnash14	22	20	37	.01	-1.790463E-01	8.09E-15	1.00E+03	1.00E+03
gnash15	22	20	39	.01	-3.546991E+02	8.54E-12	1.00E+03	1.00E+03
gnash16	22	20	52	.02	-2.414420E+02	4.51E-11	1.00E+03	1.00E+03
gnash17	22	20	24	.01	-9.074910E+01	4.46E-11	1.00E+03	1.00E+03
gnash18	22	20	21	.01	-2.569821E+01	1.17E-11	1.00E+03	1.00E+03
gnash19	22	20	21	0	-6.116708E+00	1.60E-11	1.00E+02	1.00E+02
hakonsen	14	12	28	0	2.436681E+01	1.84E-12	1.00E+02	1.00E+02
hs044-i	31	24	51	.03	1.561777E+01	1.06E-11	1.00E+01	1.00E+02
incid-set1-16	711	716	65	8	4.467425E-09	2.35E-12	1.00E+03	1.00E+03
incid-set1-32	2951	2964	353	231.73	3.058662E-07	4.56E-12	1.00E+03	1.00E+03
incid-set1-8	167	168	22	.42	7.140892E-09	2.71E-11	1.00E+02	1.00E+02
incid-set1c-16	711	731	62	7.06	3.684200E-09	1.96E-12	1.00E+03	1.00E+03
incid-set1c-32	2951	2995	243	153.58	9.952331E-06	6.68E-11	1.00E+03	1.00E+03
incid-set1c-8	167	175	26	.47	1.286690E-09	5.65E-12	1.00E+02	1.00E+02
incid-set2-16	711	716	63	7.84	3.216853E-03	1.27E-12	1.00E+03	1.00E+03
incid-set2-32	2951	2964	246	186.31	1.736491E-03	1.69E-12	1.00E+03	1.00E+03
incid-set2-8	167	168	32	.62	4.517882E-03	5.34E-11	1.00E+02	1.00E+02
incid-set2c-16	711	731	54	6.11	3.599585E-03	1.15E-12	1.00E+03	1.00E+03
incid-set2c-32	2951	2995	228	152.54	2.435879E-03	4.52E-12	1.00E+03	1.00E+03
incid-set2c-8	167	175	27	.46	5.471291E-03	1.85E-11	1.00E+02	1.00E+02
jr1	4	2	10	0	5.000000E-01	3.57E-09	1.00E+00	1.00E+00
jr2	4	2	35	0	5.000000E-01	2.60E-09	1.00E+00	1.00E+01
kth1	4	2	10	0	1.207329E-08	5.71E-09	1.00E+00	1.00E+00
kth2	4	2	9	0	2.707954E-08	2.42E-08	1.00E+00	1.00E+00
kth3	4	2	40	0	5.000000E-01	2.94E-09	1.00E+00	1.00E+01
liswet1-050	203	153	20	.04	1.399429E-02	1.31E-11	1.00E+01	1.00E+01
liswet1-100	403	303	22	.12	1.373395E-02	3.17E-12	1.00E+02	1.00E+02
liswet1-200	803	603	26	.33	1.700859E-02	4.99E-13	1.00E+02	1.00E+02

MacMPEC testset LOQO results								
problem	vars	cons	iters	time	optimal obj. fn.	final relax	initial pen	final pen
nash1	9	6	15	0	1.014401E-09	1.01E-10	1.00E+01	1.00E+01
outrata31	10	8	13	0	3.207700E+00	1.89E-10	1.00E+01	1.00E+01
outrata32	10	8	13	0	3.449404E+00	4.20E-10	1.00E+01	1.00E+01
outrata33	10	8	12	0	4.604254E+00	1.03E-09	1.00E+01	1.00E+01
outrata34	10	8	11	0	6.592684E+00	7.83E-09	1.00E+01	1.00E+01
pack-comp1-16	693	736	187	19.68	6.169515E-01	1.21E-10	1.00E+01	1.00E+01
pack-comp1-32 [†]	2917	3068	162	70.95	6.529792E-01	1.51E-09	1.00E+01	1.00E+01
pack-comp1-8	157	170	77	.99	6.000000E-01	1.47E-10	1.00E+01	1.00E+01
pack-comp1c-16 [†]	693	751	38	2.22	6.230428E-01	1.02E-10	1.00E+01	1.00E+01
pack-comp1c-32 [†]	2917	3099	108	48.74	6.614431E-01	2.35E-10	1.00E+01	1.00E+01
pack-comp1c-8	157	177	35	.41	6.000000E-01	1.75E-10	1.00E+01	1.00E+01
pack-comp1p-16	693	691	152	13.19	6.169500E-01	2.73E-14	1.00E+06	1.00E+06
pack-comp1p-32	2917	2915	386	186.85	6.529793E-01	1.59E-14	1.00E+06	1.00E+06
pack-comp1p-8 [†]	157	155	310	9.05	5.999999E-01	4.01E-16	1.00E+06	1.00E+06
pack-comp2-16	693	736	183	22.17	7.271355E-01	6.71E-10	1.00E+01	1.00E+01
pack-comp2-32 [†]	2917	3068	106	47.10	7.826040E-01	2.86E-10	1.00E+01	1.00E+01
pack-comp2-8	157	170	55	.72	6.731172E-01	1.42E-10	1.00E+01	1.00E+01
pack-comp2c-16 [†]	693	751	29	1.65	7.274676E-01	1.11E-09	1.00E+01	1.00E+01
pack-comp2c-32 [†]	2917	3099	78	30.40	7.829438E-01	1.58E-10	1.00E+01	1.00E+01
pack-comp2c-8	157	177	24	.27	6.734582E-01	3.05E-10	1.00E+01	1.00E+01
pack-comp2p-16	693	691	128	11.46	7.271354E-01	2.28E-14	1.00E+06	1.00E+06
pack-comp2p-32 [†]	2917	2915	209	83.20	7.826040E-01	1.48E-15	1.00E+06	1.00E+06
pack-comp2p-8	157	155	70	.86	6.731150E-01	8.69E-16	1.00E+06	1.00E+06
pack-rig1-16	539	537	216	13.91	8.260127E-01	1.00E-09	1.00E+01	1.00E+01
pack-rig1-32 [†]	2331	2329	216	72.57	8.508944E-01	3.40E-08	1.00E+01	1.00E+01
pack-rig1-8	120	118	51	.41	7.879318E-01	1.91E-10	1.00E+01	1.00E+01
pack-rig1c-16	539	552	104	5.64	8.264985E-01	1.87E-09	1.00E+01	1.00E+01
pack-rig1c-32 [†]	2331	2360	83	27.17	8.516407E-01	1.11E-09	1.00E+01	1.00E+01
pack-rig1c-8	120	125	28	.21	7.883002E-01	1.63E-09	1.00E+01	1.00E+01
pack-rig1p-16	649	647	42	2.16	8.260127E-01	9.17E-14	1.00E+05	1.00E+05
pack-rig1p-32 ^{††}	2717	2715	435	193.60	8.508944E-01	3.86E-08	1.00E+01	1.00E+01
pack-rig1p-8	153	151	28	.23	7.879318E-01	5.06E-14	1.00E+04	1.00E+04
pack-rig2-16 [†]	525	523			IL	1.38E-04	1.00E+01	1.00E+06
pack-rig2-32 [†]	2242	2240			IL	-5.95E+05	1.00E+01	1.00E+01
pack-rig2-8	116	114	29	.24	7.804043E-01	5.06E-10	1.00E+01	1.00E+01
pack-rig2c-16 [†]	525	538			IL	2.33E-02	1.00E+01	1.00E+01
pack-rig2c-32 [†]	2242	2271			IL	1.83E-04	1.00E+01	1.00E+01
pack-rig2c-8	116	121	29	.22	7.993058E-01	3.06E-10	1.00E+01	1.00E+01
pack-rig2p-16	631	629	51	2.92	1.085137E+00	6.79E-15	1.00E+05	1.00E+05
pack-rig2p-32 [†]	2623	2621	253	83.62	1.135885E+00	1.61E-13	1.00E+05	1.00E+05
pack-rig2p-8	149	147	29	.24	7.804043E-01	1.11E-13	1.00E+04	1.00E+04
pack-rig3-16	528	526	174	10.39	8.004305E-01	5.03E-10	1.00E+01	1.00E+01
pack-rig3-32	2248	2246	451	173.37	8.864084E-01	2.18E-09	1.00E+01	1.00E+01
pack-rig3-8	116	114	28	.23	7.352021E-01	3.13E-11	1.00E+01	1.00E+01
pack-rig3c-16	528	541	312	18.66	8.186004E-01	2.53E-10	1.00E+01	1.00E+01

MacMPEC testset LOQO results								
problem	vars	cons	iters	time	optimal obj. fn.	final relax	initial pen	final pen
pack-rig3c-32 [†]	2248	2277	99	26.57	9.089097E-01	1.00E-09	1.00E+01	1.00E+01
pack-rig3c-8	116	121	33	.26	7.534726E-01	1.39E-10	1.00E+01	1.00E+01
portfl-i-1	100	37	30	.04	1.503723E-05	3.16E-10	1.00E+01	1.00E+01
portfl-i-2	100	37	29	.04	1.457641E-05	3.78E-11	1.00E+01	1.00E+01
portfl-i-3	100	37	27	.04	6.279030E-06	1.35E-10	1.00E+01	1.00E+01
portfl-i-4	100	37	28	.04	2.186321E-06	1.26E-10	1.00E+01	1.00E+01
portfl-i-6	100	37	30	.03	2.364383E-06	5.01E-11	1.00E+01	1.00E+01
qpec1	51	40	10	0	8.000000E+01	1.36E-09	1.00E+02	1.00E+02
qpec-100-1	206	202	23	.73	9.900279E-02	1.52E-10	1.00E+02	1.00E+02
qpec-100-2	211	202	18	.76	-6.445211E+00	1.16E-10	1.00E+02	1.00E+02
qpec-100-3	211	204	26	.86	-5.481671E+00	1.26E-10	1.00E+02	1.00E+02
qpec-100-4	221	204	23	1.07	-4.058518E+00	3.62E-11	1.00E+02	1.00E+02
qpec2	51	40	108	.04	4.499392E+01	2.67E-08	1.00E+02	1.00E+05
qpec-200-1	411	404	74	14.19	-1.934830E+00	6.13E-12	1.00E+02	1.00E+03
qpec-200-2	421	404	19	3.88	-2.403634E+01	2.79E-10	1.00E+02	1.00E+02
qpec-200-3	421	408	56	10.28	-1.953406E+00	6.69E-12	1.00E+02	1.00E+03
qpec-200-4	441	408	35	11.25	-6.230523E+00	2.63E-09	1.00E+02	1.00E+02
ralph1 ^{*†}	4	2	84	0	-2.500110E-04	2.50E-07	1.00E+00	1.00E+03
ralph2 ^{††}	4	2			IL	1.57E+04	1.00E+00	1.00E+00
ralphmod	205	200	148	12.19	-5.174566E+02	2.63E-07	1.00E+03	1.00E+03
scholtes1	5	2	10	.01	2.000000E+00	1.01E-09	1.00E+01	1.00E+01
scholtes2	5	2	11	0	1.500000E+01	1.69E-09	1.00E+01	1.00E+01
scholtes3	4	2	33	0	5.000000E-01	1.84E-09	1.00E+00	1.00E+01
scholtes4 ^{*†}	5	4	96	.01	-1.000003E-03	1.00E-06	1.00E+00	1.00E+03
scholtes5	6	4	10	.01	1.000000E+00	5.05E-09	1.00E+00	1.00E+00
siouxfls	4152	4124	72	85.62	2.080014E+02	6.92E-14	1.00E+04	1.00E+04
siouxfls1 ^{††}	4152	4124			IL	-1.75E-03	1.00E+04	1.00E+04
sl1	12	8	25	0	1.000093E-04	2.08E-09	1.00E+00	1.00E+00
stackelberg1	5	3	15	0	-3.266667E-03	5.87E-10	1.00E+02	1.00E+02
tap-09	119	100	114	.33	1.102311E+02	2.71E-14	1.00E+03	1.00E+03
tap-15	278	250	55	.66	1.856721E+02	4.09E-14	1.00E+03	1.00E+03
water-FL ^{††}	258	204			IL	2.23E-07	1.00E+03	1.00E+03
water-net ^{††}	81	64			IL	4.14E-01	1.00E+02	1.00E+02

-† means the starting slack values in LOQO had to be set to .1, except for *bar-truss-3*, for which they were set to 100

-‡ means the problem was listed as infeasible in the original testset

-†† means the problem is rank-deficient, a common source of difficulty for interior-point methods

-††: the initial estimate for the penalty parameter was 1.00E+05, for which LOQO was unable to obtain a solution; the solution was obtained by manually setting the penalty at 10 and setting starting slacks to .1

-††: as described before, this problem is unbounded for penalty < 2; LOQO solves it easily if the penalty is set to be greater than 2

-*†: means LOQO had trouble obtaining accuracy on the problem as the solution it was trying to find is not a KKT point

MacMPEC: LOQO vs. FILTER				
	LOQO		FILTER	
problem	iters	optimal obj. fn.	iters	optimal obj. fn.
bard1	13	1.700000E+01	3	1.700000E+01
bard1m	13	1.700000E+01	3	1.700000E+01
bard2	36	6.598000E+03	1	6.598000E+03
bard2m	21	-6.598000E+03	1	-6.598000E+03
bard3	19	-1.267871E+01	4	-1.267871E+01
bard3m	24	-1.267871E+01	4	-1.267871E+01
bar-truss-3 [†]	29	1.016657E+04	10	1.016660E+04
bem-milanc30-s	127	9.443495E-02	62	1.298310E-01
bilev1	27	1.117272E-09		can't open
bilev2	22	-6.600000E+03	7	-6.600000E+03
bilevel1	37	3.851617E-09	2	-6.000000E+01
bilevel2	22	-6.600000E+03	7	-6.600000E+03
bilevel3	24	-1.267871E+01	7	-1.267871E+01
bilin	31	1.460000E+01	2	5.466670E+00
dempe	15	3.125001E+01	1	3.125000E+01
design-cent-1	25	1.860647E+00	4	1.860650E+00
design-cent-2	24	3.483816		INF
design-cent-3	24	3.723370E+00		rho < eps
design-cent-4	26	-2.130414E-09	4	3.079200E+00
desilva	23	-1.000000E+00	2	-1.000000E+00
df1	24	3.109960E-09	2	0.000000E+00
ex9.1.1	15	-1.300000E+01	1	-1.300000E+00
ex9.1.10	12	-3.250000E+00	1	-3.250000E+00
ex9.1.2	13	-6.250000E+00	3	-6.250000E+00
ex9.1.3	37	-2.920000E+01	3	-2.920000E+00
ex9.1.4	15	-3.700000E+01	2	-3.700000E+01
ex9.1.5	13	-1.000000E+00	3	-1.000000E+00
ex9.1.6	28	-4.900000E+01	3	-2.100000E+01
ex9.1.7	29	-2.600000E+01	3	-2.300000E+01
ex9.1.8	12	-3.250000E+00	1	-3.250000E+01
ex9.1.9	17	3.111111E+00	3	3.111110E+00
ex9.2.1	14	1.700000E+01	3	1.700000E+01
ex9.2.2 [†]	42	9.999167E+01	22	1.000000E+02
ex9.2.3	17	5.000000E+00	1	-5.500000E+01
ex9.2.4	11	5.000000E-01	3	5.000000E-01
ex9.2.5	14	9.000000E+00	7	9.000000E+00
ex9.2.6	23	-1.000000E+00	3	-1.000000E+00
ex9.2.7	14	1.700000E+01	3	1.700000E+01
ex9.2.8	3	1.500000E+00	3	1.500000E+00
ex9.2.9	13	2.000000E+00	3	2.000000E+00
flp2	13	4.150241E-09	3	0.000000E+00
flp4-1	37	5.933353E-09	3	2.77635E-30
flp4-2	33	7.093403E-10	3	0.000000E+00
flp4-3	139	1.881357E-09	3	1.191820E-30
flp4-4	65	2.386228E-08	3	5.896360E-31

MacMPEC: LOQO vs. FILTER				
	LOQO		FILTER	
problem	iters	optimal obj. fn.	iters	optimal obj. fn.
gauvin	18	2.000000E+01	3	2.000000E+01
gnash10	46	-2.308232E+02	8	-2.308230E+02
gnash11	58	-1.299119E+02	8	-1.299120E+02
gnash12	30	-3.693311E+01	9	-3.693310E+01
gnash13	45	-7.061783E+00	13	-7.061780E+00
gnash14	37	-1.790463E-01	10	-1.790460E-01
gnash15	39	-3.546991E+02	18	-3.546990E+02
gnash16	52	-2.414420E+02	16	-2.414420E+02
gnash17	24	-9.074910E+01	17	-9.074910E+01
gnash18	21	-2.569821E+01	15	-2.569820E+01
gnash19	21	-6.116708E+00	10	-6.116710E+00
hakonsen	28	2.436681E+01	10	2.436680E+01
hs044-i	51	1.561777E+01	6	1.561780E+01
incid-set1-16	65	4.467425E-09	33	1.740230E-01
incid-set1-32	353	3.058662E-07		IL
incid-set1-8	22	7.140892E-09	34	2.320310E-01
incid-set1c-16	62	3.684200E-09	34	1.740230E-01
incid-set1c-32	243	9.952331E-06	37	1.479300E-01
incid-set1c-8	26	1.286690E-09	39	2.320310E-01
incid-set2-16	63	3.216853E-03	19	3.037130E-03
incid-set2-32	246	1.736491E-03	114	1.750980E-03
incid-set2-8	32	4.517882E-03	48	4.517880E-03
incid-set2c-16	54	3.599585E-03	37	3.678050E-03
incid-set2c-32	228	2.435879E-03	41	2.451130E-03
incid-set2c-8	27	5.471291E-03	24	5.471270E-03
jr1	10	5.000000E-01	1	5.000000E-01
jr2	35	5.000000E-01	7	5.000000E-01
kth1	10	1.207329E-08	1	0.000000E+00
kth2	9	2.707954E-08	2	0.000000E+00
kth3	40	5.000000E-01	4	5.000000E-01
liswet1-050	20	1.399429E-02	1	1.399430E-02
liswet1-100	22	1.373395E-02	1	1.373390E-02
liswet1-200	26	1.700859E-02	1	1.700860E-02
nash1	15	1.014401E-09	5	0.000000E+00
outrata31	13	3.207700E+00	8	3.207700E+00
outrata32	13	3.449404E+00	8	3.449400E+00
outrata33	12	4.604254E+00	7	4.604250E+00
outrata34	11	6.592684E+00	6	6.592680E+00
pack-comp1-16	187	6.169515E-01	20	6.169510E-01
pack-comp1-32 [†]	162	6.529792E-01	21	6.529790E-01
pack-comp1-8	77	6.000000E-01	8	6.000000E-01
pack-comp1c-16 [†]	38	6.230428E-01	5	6.230430E-01
pack-comp1c-32 [†]	108	6.614431E-01	13	6.614430E-01
pack-comp1c-8	35	6.000000E-01	8	6.000000E-01
pack-comp1p-16	152	6.169500E-01	45	6.169510E-01
pack-comp1p-32	386	6.529793E-01	42	6.529770E-01
pack-comp1p-8 [†]	310	5.999999E-01	53	6.000000E-01
pack-comp2-16	183	7.271355E-01	43	7.271350E-01
pack-comp2-32 [†]	106	7.826040E-01	78	7.826040E-01
pack-comp2-8	55	6.731172E-01	8	6.731170E-01
pack-comp2c-16 [†]	29	7.274676E-01	15	7.274680E-01
pack-comp2c-32 [†]	78	7.829438E-01	7	7.829440E-01

MacMPEC: LOQO vs. FILTER				
problem	LOQO		FILTER	
	iters	optimal obj. fn.	iters	optimal obj. fn.
pack-comp2c-8	24	6.734582E-01	6	6.734580E-01
pack-comp2p-16	128	7.271354E-01	32	7.271350E-01
pack-comp2p-32 [†]	209	7.826040E-01	47	7.826040E-01
pack-comp2p-8	70	6.731150E-01	60	6.731170E-01
pack-rig1-16	216	8.260127E-01	64	8.260130E-01
pack-rig1-32 [†]	216	8.508944E-01	19	8.508950E-01
pack-rig1-8	51	7.879318E-01	7	7.879320E-01
pack-rig1c-16	104	8.264985E-01	11	8.264980E-01
pack-rig1c-32 [†]	83	8.516407E-01	18	8.516410E-01
pack-rig1c-8	28	7.883002E-01	6	7.883000E-01
pack-rig1p-16	42	8.260127E-01	28	2.644980E+02
pack-rig1p-32 ^{††}	435	8.508944E-01	125	2.242060E+03
pack-rig1p-8	28	7.879318E-01	14	3.594420E+01
pack-rig2-16 [‡]		IL		INF
pack-rig2-32 [‡]		IL		INF
pack-rig2-8	29	7.804043E-01	10	7.804040E-01
pack-rig2c-16 [‡]		IL	6	1.000000E+00
pack-rig2c-32 [‡]		IL	11	1.000000E+00
pack-rig2c-8	29	7.993058E-01	6	7.993060E-01
pack-rig2p-16	51	1.085137E+00	10	6.259330E+02
pack-rig2p-32 [†]	253	1.135885E+00	20	8.717450E+02
pack-rig2p-8	29	7.804043E-01	20	4.667880E+01
pack-rig3-16	174	8.004305E-01	32	8.004310E-01
pack-rig3-32	451	8.864084E-01		time limit
pack-rig3-8	28	7.352021E-01	10	7.352020E-01
pack-rig3c-16	312	8.186004E-01	9	8.186000E-01
pack-rig3c-32 [†]	99	9.089097E-01	14	9.089100E-01
pack-rig3c-8	33	7.534726E-01	7	7.534730E-01
portfl-i-1	30	1.503723E-05	5	1.502420E-05
portfl-i-2	29	1.457641E-05	4	1.457280E-05
portfl-i-3	27	6.279030E-06	4	6.26499E-06
portfl-i-4	28	2.186321E-06	4	2.177340E-06
portfl-i-6	30	2.364383E-06	4	2.361330E-06
qpec1	10	8.000000E+01	3	8.000000E+01
qpec-100-1	23	9.900279E-02	7	9.900280E-02
qpec-100-2	18	-6.445211E+00	7	-6.260490E+00
qpec-100-3	26	-5.481671E+00	6	-5.482870E+00
qpec-100-4	23	-4.058518E+00	5	-3.672010E+00
qpec2	108	4.499392E+01	2	4.500000E+01
qpec-200-1	74	-1.934830E+00	10	-1.934830E+00
qpec-200-2	19	-2.403634E+01	10	-2.407740E+01
qpec-200-3	56	-1.953406E+00	11	-1.953410E+00
qpec-200-4	35	-6.230523E+00	5	-6.193230E+00
ralph1 ^{*†}	84	-2.500110E-04	27	-6.860340E-07
ralph2 ^{††}		IL	11	-6.256730E-07
ralphmod	148	-5.174566E+02	7	-5.025770E+02
scholtes1	10	2.000000E+00	4	2.000000E+00
scholtes2	11	1.500000E+01	2	1.500000E+01
scholtes3	33	5.000000E-01	4	5.000000E-01
scholtes4 ^{*†}	96	-1.000003E-03	26	-2.541610E-07
scholtes5	10	1.000000E+00	1	1.000000E+00
siouxfls	72	2.080014E+02	10	2.082580E+02
siouxfls1 ^{††}		IL		inf QP
sl1	25	1.000093E-04	1	1.000000E-04
stackelberg1	15	-3.266667E-03	4	-3.266670E+03

MacMPEC: LOQO vs. FILTER				
problem	LOQO		FILTER	
	iters	optimal obj. fn.	iters	optimal obj. fn.
tap-09	114	1.102311E+02	21	1.091310E+02
tap-15	55	1.856721E+02	28	1.876320E+02
water-FL ^{†‡}		IL	277	3.475600E+03
water-net ^{†‡}		IL		time limit

In addition to the problem documented in Section 5, the penalty method was able to solve some problems which a standard implementation of LOQO was unable to solve. These problems were *ex9.2.2*, *pack-rig2-16*, *pack-rig2p-32*, *qpec2*, *ralph1*, *scholtes4*, *tap-09*, and *tap-15*.

B.1. Ferris Problems (pure complementarity problems).

problem	loqo result	comment	filter result
cammcp	fail	non-differentiable (see below)	fail
duopoly	solved		solved
ehl.kost	solved	when initial slacks set to .01	solved
electric	error	error given by AMPL during presolve	AMPL error
forcedsa	solved		fail
games	solved		fail
lincont	solved	when all variables set to 1 (see below)	solved
pgvon105	fail	non-differentiable (see below)	fail
shubik	solved		solved
simple-ex	solved	when variables $x_1 - x_4$ initialized at 1 (see below)	solved
spillmcp	solved		solved

- *cammcp*, *pgvon105*: These two problems have functions which are not differentiable at a point, a known source of trouble for interior-point methods. Hence, LOQO's failure on these two problems does not seem to be due to their being equilibrium problems as such.
- *ehl.kost*, *simple-ex*: If nonnegative variables are initialized at 0 or not initialized at all (in which case LOQO automatically sets them to 0), an interior-point method can have trouble. It is therefore understandable that LOQO's success on these problems depends on the variables in question being initialized away from 0.

HANDE Y. BENSON, DECISION SCIENCES DEPARTMENT, LeBow College of Business, Drexel University, Philadelphia, PA 19104, *E-mail address*, BENSON@DREXEL.EDU: AND

ARUN SEN, DEPT. OF OPERATIONS RESEARCH AND FINANCIAL ENGINEERING, PRINCETON UNIVERSITY, PRINCETON, NJ 08544, *E-mail address*, ASEN@PRINCETON.EDU: AND

DAVID F. SHANNO, RUTCOR - RUTGERS CENTER OF OPERATIONS RESEARCH, RUTGERS UNIVERSITY, NEW BRUNSWICK, NJ 08903, *E-mail address*, SHANNO@RUTCOR.RUTGERS.EDU: AND

ROBERT J. VANDERBEI, DEPT. OF OPERATIONS RESEARCH AND FINANCIAL ENGINEERING, PRINCETON UNIVERSITY, PRINCETON, NJ 08544, *E-mail address*, RVDB@PRINCETON.EDU: